Name: Upol Saipidin kyzy Neptun ID: IHLG15 Assignment: 3 Task number: 2

Task description:

Snake

We have a rattlesnake in a desert, and our snake is initially two units long (head and rattler). We have to collect with our snake the foods on the level, that appears randomly. Only one food piece is placed randomly at a time on the level (on a field, where there is no snake). The snake starts off from the center of the level in a random direction. The player can control the movement of the snake's head with keyboard buttons. If the snake eats a food piece, then its length grow by one unit.

It makes the game harder that there are rocks in the desert. If the snake collides with a rock, then the game ends. We also lose the game, if the snake goes into itself, or into the boundary of the game level. In these situations show a popup messagebox, where the player can type his name and save it together with the amount of food eaten to the database. Create a menu item, which displays a highscore table of the players for the 10 best scores. Also, create a menu item which restarts the game.

Description of each method:

SnakePanel.java class:

setToRadomDirection(): in the beginning of each level, it sets direction of the snake randomly;

restart(): restarts the game;

startGame(): starts the game;

paintComponent(Graphics g): draws components of the GUI component;

doesCollideWithSnake(): checks whether an apple or rocks collide with the snake;

doesCollideWithRocks(): checks whether the snake or apple collides with rocks;

generateObstacles(): creates rocks;

draw(Graphics g): draws elements of the game;

newApple(): creates new apple randomly when score increases;

move(): does movement of the snake;

checkApple(): checks an apple, if snake eats an apple, score increases by one and level of the game creates here;

checkCollision(): checks collision of the snake, if it eats itself or crosses the border or collides rocks;

gameOver(): checks lose of game and shows dialog to enter the name;

actionPerformed(ActionEvent e): action performance while game is running;

keyPressed(KeyEvent e): shows the event when key is pressed(WASD).

MenuGUI.java

paintComponent(Graphics g): draw image of the given picture;

HighScore.java

getName(): returns name of the player;

getScore(): returns score of the player;

compareByScore(HighScore hs1, HighScore hs2): compares two player's scores and returns the best score.

getHighScores(): firstly executes sql query and adds players into list and returns them sorted;

insertScore(String name, int score): inserts player which is given as a parameter into the database;

putHighScore(String pName, int score): gets sorted list of players and compares its size with the max number of of players, if it is less, then inserts a player, otherwise checks last player of the list and new player's score and inserts higher score and deletes another player;

sortByScore(ArrayList<HighScore> highScores): sorts list of players by their score;

deleteScores(int score): deletes score and modifies the table;

getColumnNamesArray(): returns ID, Name, Score of the player;

getDataTable(): creates table of the players in matrix format;

SnakeGameGUI.java

addEventListeners(JMenuItem exit , JMenuItem restart, JButton newGameButton, JButton scoresButton):

  exit.addActionListener(new ActionListener(): exits the game;

  restart.addActionListener(new ActionListener(): starts the new game(menu);

  newGameButton.addActionListener(new ActionListener(): starts the new game(button);

  scoresButton.addActionListener(new ActionListener(): opens score's frame and shows best players;

createHighScoresFrame(): creates well-designed from for the high scores table;

createGameFrame(): creates a frame for playing.

UML Class Diagram: